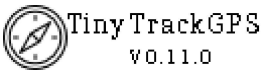
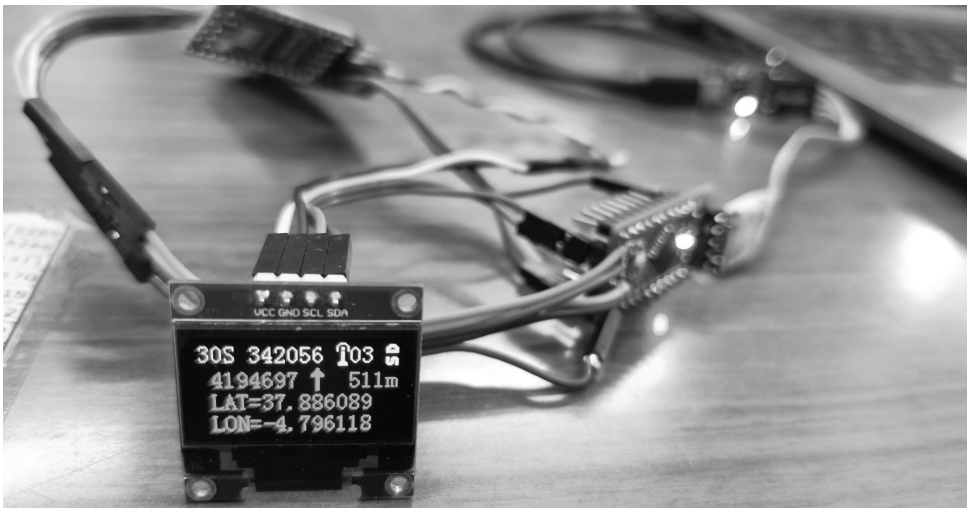
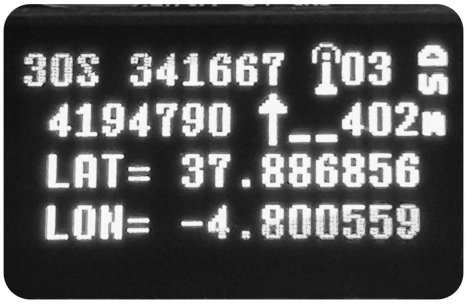
TinyTrackGPS



A simple track GPS to SD card logger.





This program is written in C/C++ for Arduino © UNO R3 and other compatible microcontrollers based on Atmega328 and similar.

It is tested on:

- UNO R3 board (Arduino UNO compatible board based on Atmega328).
- ProMini 5v 16MHz (Arduino ProMini compatible board based on Atmega328p).
- Lgt8f328p (a replacement Arduino Pro Mini).Tested v0.1, v0.2 and since v0.10. (default option)

List of componets

This project use components list above:

- Arduino © UNO board or equivalent AVR.
- NMEA 6,8 GPS module.
- MicroSD module and card.
- LCD 16×2 char display module (wired or I2C), or OLED 0.96" I2C (SSD1306) (Optional).

I use this components:

- LGT8F328P LQFP-32 MiniEVB 32MHz board.
- U-blox NEO-M8N GPS module. A concurrent reception module up to 3 GNSS (GPS, Galileo, GLONASS, BeiDou)
- MicroSD module with MicroSD 4Gb Card FAT32 formatted.
- SSD1306 0.96" 128x64 OLED I2C display module.

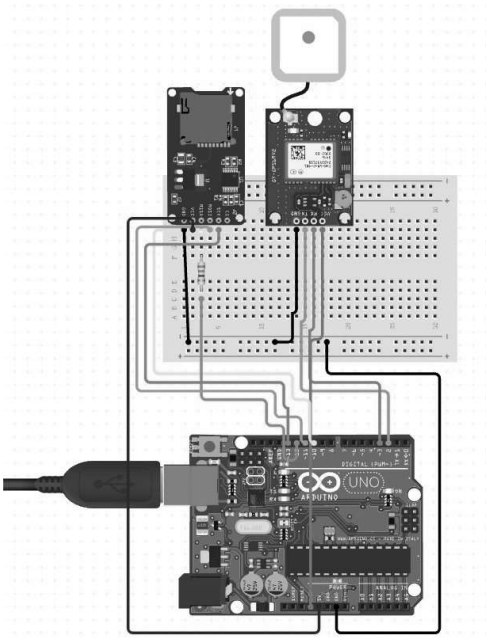
Additional:

- Lipo 3,7V 900mAh with protect.
- MicroUsb charge module.

NO DISPLAY

Now you can use a minimal hardware to track GPS location. When no display use MicroSD module and card are mandatory. Comment all lines in 'config.h' file:

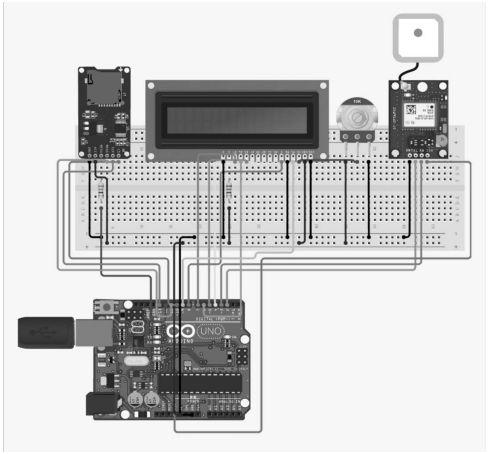
```
//#define DISPLAY_TYPE_SDD1306_128X64     // Para usar pantalla OLED 0.96" I2C
128x64 pixels
//#define DISPLAY_TYPE_SDD1306_128X64_lcdgfx // Para usar pantalla OLED 0.96" I2C
128x64 pixels (lcdgfx library)
//#define DISPLAY_TYPE_LCD_16X2           // Para usar LCD 16 x 2 carateres.
//#define DISPLAY_TYPE_LCD_16X2_I2C       // Para usar LCD 16 x 2 carateres. I2C.
```



LCD 16x2

If you use LCD 16x2 char wired (6-wires), uncomment line like this in 'config.h' file:

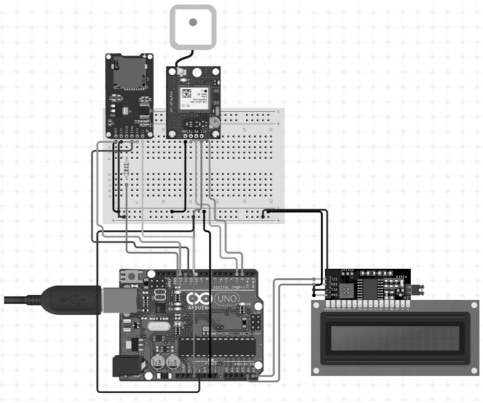
```
#define DISPLAY_TYPE_LCD_16X2
```



LCD 16x2 I2C

If you use LCD 16x2 char I2C (4-wires), uncomment line like this in 'config.h' file:

```
#define DISPLAY_TYPE_LCD_16X2_I2C
```



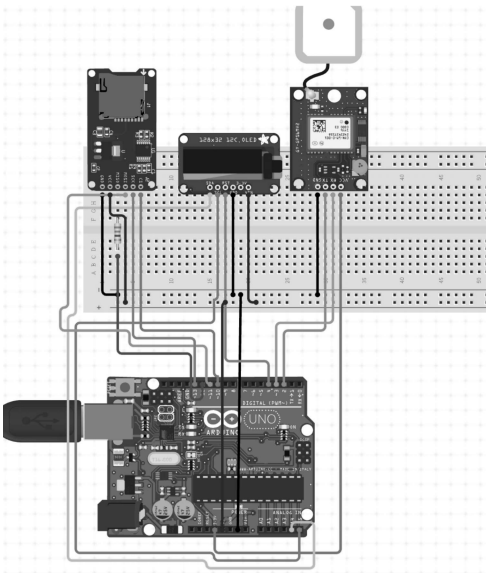
OLED 0'96" 128x64 I2C

If you use OLED 0'96" 128X64 I2C (4-wires), uncomment line like this in 'config.h' file:

```
#define DISPLAY_TYPE_SDD1306_128X64 // Uses u8g2 library.
```

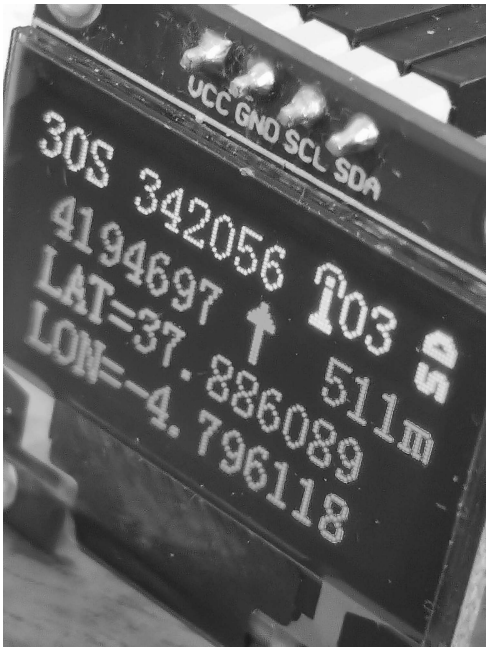
-- or --

```
#define DISPLAY_TYPE_SDD1306_128X64_lcdgfx // Uses lcdgfx library.
```

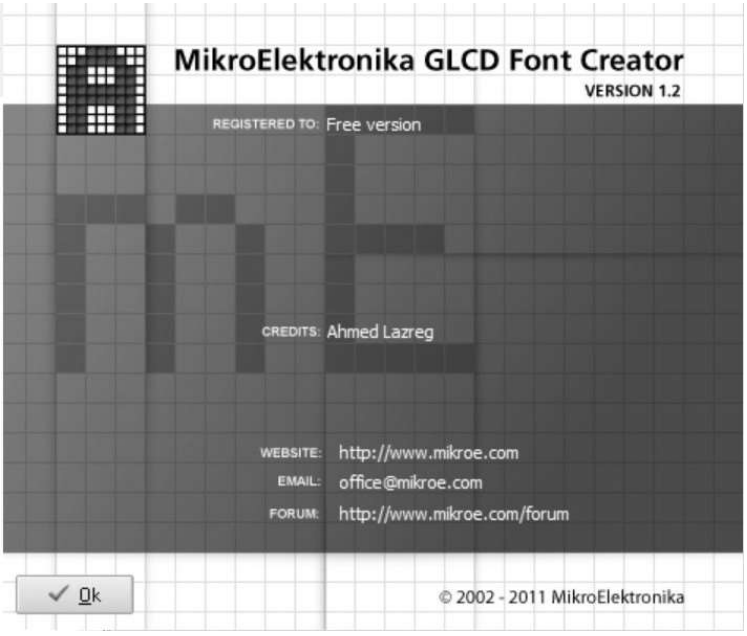


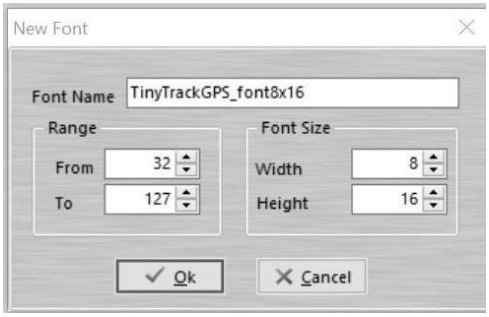
Lcdgfx library.

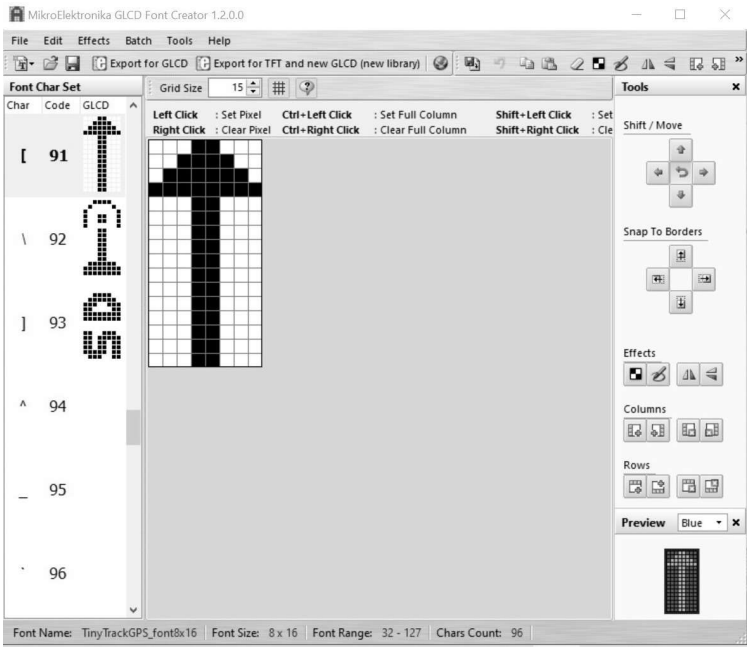
TinyTrackGPS uses this library by default. Use less flash memory and RAM. Fast running and no display flickering. For more information see at https://github.com/lexus2k/lcdgfx.

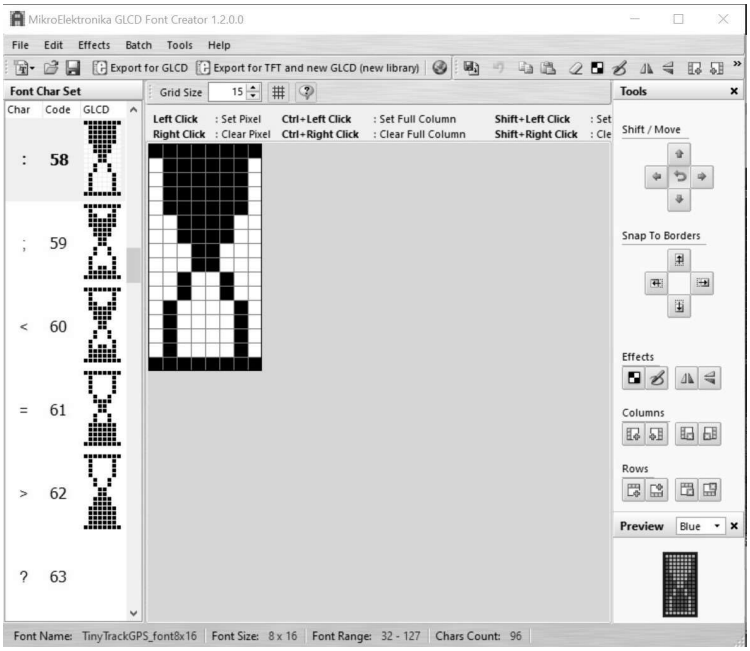


The project define a new font (TinyTrackGPS_font8x16), a modified version of ssd1306xled_font8x16 of lcdgfx fonts (canvas/fonts/fonts.c). For information about create a new font visit: https://github.com/lexus2k/lcdgfx/wiki/How-to-create-new-font-for-the-library.






MikroElektronika GLCD Font Creator 1.2.0.0
File Edit Effects Batch Tools Help
Export for GLCD
Export for TFT and new GLCD (new library)
Font Char Set
Char Code GLCD
[91
\ 92
] 93
^ 94
_ 95
` 96
Grid Size 15
Left Click : Set Pixel
Right Click : Clear Pixel
Ctrl+Left Click : Set Full Column
Ctrl+Right Click : Clear Full Column
Shift+Left Click
Shift+Right Click
Tools
Shift / Move
Snap To Borders
Effects
Columns
Rows
Preview
Blue
Font Name: TinyTrackGPS_font8x16
Font Size: 8 x 16
Font Range: 32 - 127
Chars Count: 96


MikroElektronika GLCD Font Creator 1.2.0.0
File Edit Effects Batch Tools Help
Export for GLCD
Export for TFT and new GLCD (new library)
Font Char Set
Char Code GLCD
: 58
; 59
< 60
= 61
> 62
? 63
Grid Size 15
Left Click : Set Pixel
Right Click : Clear Pixel
Ctrl+Left Click : Set Full Column
Ctrl+Right Click : Clear Full Column
Shift+Left Click
Shift+Right Click
Tools
Shift / Move
Snap To Borders
Effects
Columns
Rows
Preview
Blue
Font Name: TinyTrackGPS_font8x16
Font Size: 8 x 16
Font Range: 32 - 127
Chars Count: 96

```
const PROGMEM uint8_t TinyTrackGPS_font8x16[] = {
  0x00,// 0x00 means fixed font type - the only supported by the library
  0x08,// 0x08 = 8 - font width in pixels
  0x10,// 0x10 = 16 - font height in pixels
  0x2d,// Start char.
  // Chars numbers and signs.
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,0x01,0x01,0x01,0x01,0x01,0x01,// - 45
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x00,0x00,0x00,0x00,0x00,// . 46
  0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x00,0x04,0x04,0x04,0x04,0x04,0x04,0x04,0x00,
// '/'->'=' 47
  0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x0F,0x10,0x20,0x20,0x10,0x0F,0x00,// 0 48
  0x00,0x10,0x10,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,// 1 49
  0x00,0x70,0x08,0x08,0x08,0x88,0x70,0x00,0x00,0x30,0x28,0x24,0x22,0x21,0x30,0x00,// 2 50
  0x00,0x30,0x08,0x88,0x88,0x48,0x30,0x00,0x00,0x18,0x20,0x20,0x20,0x11,0x0E,0x00,// 3 51
  0x00,0x00,0xC0,0x20,0x10,0xF8,0x00,0x00,0x00,0x07,0x04,0x24,0x24,0x3F,0x24,0x00,// 4 52
  0x00,0xF8,0x08,0x88,0x88,0x08,0x08,0x00,0x00,0x19,0x21,0x20,0x20,0x11,0x0E,0x00,// 5 53
  0x00,0xE0,0x10,0x88,0x88,0x18,0x00,0x00,0x00,0x0F,0x11,0x20,0x20,0x11,0x0E,0x00,// 6 54
  0x00,0x38,0x08,0x08,0xC8,0x38,0x08,0x00,0x00,0x00,0x00,0x3F,0x00,0x00,0x00,0x00,// 7 55
  0x00,0x70,0x88,0x08,0x08,0x88,0x70,0x00,0x00,0x1C,0x22,0x21,0x21,0x22,0x1C,0x00,// 8 56
  0x00,0xE0,0x10,0x08,0x08,0x10,0xE0,0x00,0x00,0x00,0x31,0x22,0x22,0x11,0x0F,0x00,// 9 57
  // Chars for wait animation.
  0x01,0x1f,0x7f,0xff,0xff,0x7f,0x1f,0x01,0x80,0xf8,0x86,0x81,0x81,0x86,0xf8,0x80,
// ':'->wait1 58
  0x01,0x1f,0x7d,0xf9,0xf9,0x7d,0x1f,0x01,0x80,0xf8,0xc6,0xe1,0xe1,0xc6,0xf8,0x80,
// ';'->wait2 59
  0x01,0x1f,0x79,0xf1,0xf1,0x79,0x1f,0x01,0x80,0xf8,0xe6,0xf1,0xf1,0xe6,0xf8,0x80,
// '<'->wait3 60
  0x01,0x1f,0x71,0xe1,0xe1,0x71,0x1f,0x01,0x80,0xf8,0xfe,0xf9,0xf9,0xfe,0xf8,0x80,
// '='->wait4 61
  0x01,0x1f,0x61,0x81,0x81,0x61,0x1f,0x01,0x80,0xf8,0xfe,0xff,0xff,0xfe,0xf8,0x80,
// '>'->wait5 62
  // Chars for display space (' ') and 'm' char.
  0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
// ? ->' ' 63
  0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x20,0x3F,0x20,0x00,0x3F,0x20,0x00,0x3F,
// @ ->'m' 64
  // Letters chars for UTM Zone.
  0x00,0x00,0xC0,0x38,0xE0,0x00,0x00,0x00,0x20,0x3C,0x23,0x02,0x02,0x27,0x38,0x20,// A 33
  0x08,0xF8,0x88,0x88,0x88,0x70,0x00,0x00,0x20,0x3F,0x20,0x20,0x20,0x11,0x0E,0x00,// B 34
  0xC0,0x30,0x08,0x08,0x08,0x08,0x38,0x00,0x07,0x18,0x20,0x20,0x20,0x10,0x08,0x00,// C 35
  0x08,0xF8,0x08,0x08,0x08,0x10,0xE0,0x00,0x20,0x3F,0x20,0x20,0x20,0x10,0x0F,0x00,// D 36
  0x08,0xF8,0x88,0x88,0xE8,0x08,0x10,0x00,0x20,0x3F,0x20,0x20,0x23,0x20,0x18,0x00,// E 37
  0x08,0xF8,0x88,0x88,0xE8,0x08,0x10,0x00,0x20,0x3F,0x20,0x00,0x03,0x00,0x00,0x00,// F 38
  0xC0,0x30,0x08,0x08,0x08,0x38,0x00,0x00,0x07,0x18,0x20,0x20,0x22,0x1E,0x02,0x00,// G 39
  0x08,0xF8,0x08,0x00,0x00,0x08,0xF8,0x08,0x20,0x3F,0x21,0x01,0x01,0x21,0x3F,0x20,// H 40
  0x00,0x08,0x08,0xF8,0x08,0x08,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,// I 41
  0x00,0x00,0x08,0x08,0xF8,0x08,0x08,0x00,0xC0,0x80,0x80,0x80,0x7F,0x00,0x00,0x00,// J 42
  0x08,0xF8,0x88,0xC0,0x28,0x18,0x08,0x00,0x20,0x3F,0x20,0x01,0x26,0x38,0x20,0x00,// K 43
  0x08,0xF8,0x08,0x00,0x00,0x00,0x00,0x00,0x20,0x3F,0x20,0x20,0x20,0x20,0x30,0x00,// L 44
  0x08,0xF8,0xF8,0x00,0xF8,0xF8,0x08,0x00,0x20,0x3F,0x00,0x3F,0x00,0x3F,0x20,0x00,// M 45
  0x08,0xF8,0x30,0xC0,0x00,0x08,0xF8,0x08,0x20,0x3F,0x20,0x00,0x07,0x18,0x3F,0x00,// N 46
  0xE0,0x10,0x08,0x08,0x08,0x10,0xE0,0x00,0x0F,0x10,0x20,0x20,0x20,0x10,0x0F,0x00,// O 47
  0x08,0xF8,0x08,0x08,0x08,0x08,0xF0,0x00,0x20,0x3F,0x21,0x01,0x01,0x01,0x00,0x00,// P 48
  0xE0,0x10,0x08,0x08,0x08,0x10,0xE0,0x00,0x0F,0x18,0x24,0x24,0x38,0x50,0x4F,0x00,// Q 49
  0x08,0xF8,0x88,0x88,0x88,0x88,0x70,0x00,0x20,0x3F,0x20,0x00,0x03,0x0C,0x30,0x20,// R 50
  0x00,0x70,0x88,0x08,0x08,0x08,0x38,0x00,0x00,0x38,0x20,0x21,0x21,0x22,0x1C,0x00,// S 51
  0x18,0x08,0x08,0xF8,0x08,0x08,0x18,0x00,0x00,0x00,0x20,0x3F,0x20,0x00,0x00,0x00,// T 52
  0x08,0xF8,0x08,0x00,0x00,0x08,0xF8,0x08,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,// U 53
  0x08,0x78,0x88,0x00,0x00,0xC8,0x38,0x08,0x00,0x00,0x07,0x38,0x0E,0x01,0x00,0x00,// V 54
  0xF8,0x08,0x00,0xF8,0x00,0x08,0xF8,0x00,0x03,0x3C,0x07,0x00,0x07,0x3C,0x03,0x00,// W 55
  0x08,0x18,0x68,0x80,0x80,0x68,0x18,0x08,0x20,0x30,0x2C,0x03,0x03,0x2C,0x30,0x20,// X 56
  0x08,0x38,0xC8,0x00,0xC8,0x38,0x08,0x00,0x00,0x00,0x20,0x3F,0x20,0x00,0x00,0x00,// Y 57
  0x10,0x08,0x08,0x08,0xC8,0x38,0x08,0x00,0x20,0x38,0x26,0x21,0x20,0x20,0x18,0x00,// Z 58
  // Personalized simbols for display info.
  0x30,0x38,0x3c,0xff,0xff,0x3c,0x38,0x30,0x00,0x00,0x00,0xff,0xff,0x00,0x00,0x00,
```

```
// '['->altitud 91
  0x3c,0x02,0x01,0xd9,0xd9,0x01,0x02,0x3c,0x00,0xc0,0xe0,0xff,0xff,0xe0,0xc0,0x00,
// '\'->antena 92
  0x78,0x7c,0x6e,0x66,0x66,0x6e,0x7c,0x78,0x7c,0xfc,0xc0,0xf8,0x7c,0x0c,0xfc,0xf8,
// ']'->sd 93
  /*  Unused letters
  0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x19,0x24,0x22,0x22,0x22,0x3F,0x20,// a 65
  0x08,0xF8,0x00,0x80,0x80,0x00,0x00,0x00,0x00,0x3F,0x11,0x20,0x20,0x11,0x0E,0x00,// b 66
  0x00,0x00,0x00,0x80,0x80,0x80,0x00,0x00,0x00,0x0E,0x11,0x20,0x20,0x20,0x11,0x00,// c 67
  0x00,0x00,0x00,0x80,0x80,0x88,0xF8,0x00,0x00,0x0E,0x11,0x20,0x20,0x10,0x3F,0x20,// d 68
  0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x1F,0x22,0x22,0x22,0x22,0x13,0x00,// e 69
  0x00,0x80,0x80,0xF0,0x88,0x88,0x88,0x18,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,// f 70
  0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x6B,0x94,0x94,0x94,0x93,0x60,0x00,// g 71
  0x08,0xF8,0x00,0x80,0x80,0x80,0x00,0x00,0x20,0x3F,0x21,0x00,0x00,0x20,0x3F,0x20,// h 72
  0x00,0x80,0x98,0x98,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,// i 73
  0x00,0x00,0x00,0x80,0x98,0x98,0x00,0x00,0x00,0xC0,0x80,0x80,0x80,0x7F,0x00,0x00,// j 74
  0x08,0xF8,0x00,0x00,0x80,0x80,0x80,0x00,0x20,0x3F,0x24,0x02,0x2D,0x30,0x20,0x00,// k 75
  0x00,0x08,0x08,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,0x20,0x20,0x00,0x00,// l 76
  0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x20,0x3F,0x20,0x00,0x3F,0x20,0x00,0x3F,// m 77
  0x80,0x80,0x00,0x80,0x80,0x80,0x00,0x00,0x20,0x3F,0x21,0x00,0x00,0x20,0x3F,0x20,// n 78
  0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x00,0x1F,0x20,0x20,0x20,0x20,0x1F,0x00,// o 79
  0x80,0x80,0x00,0x80,0x80,0x00,0x00,0x00,0x80,0xFF,0xA1,0x20,0x20,0x11,0x0E,0x00,// p 80
  0x00,0x00,0x00,0x80,0x80,0x80,0x80,0x00,0x00,0x0E,0x11,0x20,0x20,0xA0,0xFF,0x80,// q 81
  0x80,0x80,0x80,0x00,0x80,0x80,0x80,0x00,0x20,0x20,0x3F,0x21,0x20,0x00,0x01,0x00,// r 82
  0x00,0x00,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x33,0x24,0x24,0x24,0x24,0x19,0x00,// s 83
  0x00,0x80,0x80,0xE0,0x80,0x80,0x00,0x00,0x00,0x00,0x00,0x1F,0x20,0x20,0x00,0x00,// t 84
  0x80,0x80,0x00,0x00,0x00,0x80,0x80,0x00,0x00,0x1F,0x20,0x20,0x20,0x10,0x3F,0x20,// u 85
  0x80,0x80,0x80,0x00,0x00,0x80,0x80,0x80,0x00,0x01,0x0E,0x30,0x08,0x06,0x01,0x00,// v 86
  0x80,0x80,0x00,0x80,0x00,0x80,0x80,0x80,0x0F,0x30,0x0C,0x03,0x0C,0x30,0x0F,0x00,// w 87
  0x00,0x80,0x80,0x00,0x80,0x80,0x80,0x00,0x00,0x20,0x31,0x2E,0x0E,0x31,0x20,0x00,// x 88
  0x80,0x80,0x80,0x00,0x00,0x80,0x80,0x80,0x80,0x81,0x8E,0x70,0x18,0x06,0x01,0x00,// y 89
  0x00,0x80,0x80,0x80,0x80,0x80,0x80,0x00,0x00,0x21,0x30,0x2C,0x22,0x21,0x30,0x00,// z 90
*/
};
```

UST/UT Time.

(Universal Summer Timer/Universal Standard Time)

Now TinyTrackGPS record the info in local time. It is used Timezone library for that. Select the proper config at *line 111* of `TinyTrackGPS.cpp` file. There are some info for time zone:

```
// Australia Eastern Time Zone (Sydney, Melbourne)
TimeChangeRule aEDT = {"AEDT", First, Sun, Oct, 2, 660};    // UTC + 11 hours
TimeChangeRule aEST = {"AEST", First, Sun, Apr, 3, 600};    // UTC + 10 hours
Timezone ausET(aEDT, aEST);

// Moscow Standard Time (MSK, does not observe DST)
TimeChangeRule msk = {"MSK", Last, Sun, Mar, 1, 180};
Timezone tzMSK(msk);

// Central European Time (Frankfurt, Paris)
TimeChangeRule CEST = {"CEST", Last, Sun, Mar, 2, 120};     // Central European
```

```
Summer Time
TimeChangeRule CET = {"CET ", Last, Sun, Oct, 3, 60};       // Central European
Standard Time
Timezone CE(CEST, CET);

// United Kingdom (London, Belfast)
TimeChangeRule BST = {"BST", Last, Sun, Mar, 1, 60};        // British Summer Time
TimeChangeRule GMT = {"GMT", Last, Sun, Oct, 2, 0};         // Standard Time
Timezone UK(BST, GMT);

// UTC
TimeChangeRule utcRule = {"UTC", Last, Sun, Mar, 1, 0};     // UTC
Timezone UTC(utcRule);

// US Eastern Time Zone (New York, Detroit)
TimeChangeRule usEDT = {"EDT", Second, Sun, Mar, 2, -240};  // Eastern Daylight
Time = UTC - 4 hours
TimeChangeRule usEST = {"EST", First, Sun, Nov, 2, -300};   // Eastern Standard
Time = UTC - 5 hours
Timezone usET(usEDT, usEST);

// US Central Time Zone (Chicago, Houston)
TimeChangeRule usCDT = {"CDT", Second, Sun, Mar, 2, -300};
TimeChangeRule usCST = {"CST", First, Sun, Nov, 2, -360};
Timezone usCT(usCDT, usCST);

// US Mountain Time Zone (Denver, Salt Lake City)
TimeChangeRule usMDT = {"MDT", Second, Sun, Mar, 2, -360};
TimeChangeRule usMST = {"MST", First, Sun, Nov, 2, -420};
Timezone usMT(usMDT, usMST);

// Arizona is US Mountain Time Zone but does not use DST
Timezone usAZ(usMST);

// US Pacific Time Zone (Las Vegas, Los Angeles)
TimeChangeRule usPDT = {"PDT", Second, Sun, Mar, 2, -420};
TimeChangeRule usPST = {"PST", First, Sun, Nov, 2, -480};
Timezone usPT(usPDT, usPST);
```

Source

TinyTrackGPS is free software, see **License** section for more information. The code is based and get parts of the libraries above:

- TinyGPS library fork, Paul Stoffregen (https://github.com/PaulStoffregen/TinyGPS). Fixed version on 'lib'.

- SdFat library, Bill Greiman (https://github.com/greiman/SdFat). Fixed version on 'lib'.
- Lcdgfx library,Aleksei (https://github.com/lexus2k/lcdgfx).
- U8g2 library, oliver (https://github.com/olikraus/u8g2).
- Low-Power library, Rocket Scream Electronics (https://github.com/rocketscream/Low-Power).
- LiquidCrystal library, Arduino Standard Libraries (Arduino IDE).
- LiquidCrystal I2C library, John Rickman (https://github.com/johnrickman/LiquidCrystal_I2C).
- UTMConversion library, Rafael Reyes (https://github.com/RafaelReyesCarmona/UTMConversion).
- Timezone library, Jack Christensen (https://github.com/JChristensen/Timezone).
- Time library, Paul Stoffregen (https://github.com/PaulStoffregen/Time).

How to compile

Config

Edit 'config.h' file before, to configure display type uncommenting the proper line:

```
// Descomentar solo uno de los displays utilizados.
//#define DISPLAY_TYPE_SDD1306_128X64     // Para usar pantalla OLED 0.96" I2C
128x64 pixels
#define DISPLAY_TYPE_SDD1306_128X64_lcdgfx // Para usar pantalla OLED 0.96" I2C
128x64 pixels (lcdgfx library)
//#define DISPLAY_TYPE_LCD_16X2             // Para usar LCD 16 x 2 carateres.
//#define DISPLAY_TYPE_LCD_16X2_I2C       // Para usar LCD 16 x 2 carateres. I2C.
```

Modify Arduino pin where you connect the LCD 16x2 char:

```
// Definiciones para display LCD 16x2 caracteres.
#define RS 2
#define ENABLE 3
#define D0 4
#define D1 5
#define D2 6
#define D3 7
```

Modify I2C port for LCD 16x2 I2C: (connect in SCL and SDA pins)

```
// Define direccion I2C para LCD16x2 char.
#define I2C 0x27
```

Coding TimeChangeRules

Normally these will be coded in pairs for a given time zone: One rule to describe when daylight (summer) time starts, and one to describe when standard time starts.

As an example, here in the Eastern US time zone, Eastern Daylight Time (EDT) starts on the 2nd Sunday in March at 02:00 local time. Eastern Standard Time (EST) starts on the 1st Sunday in November at 02:00 local time.

Define a **TimeChangeRule** as follows:

```
TimeChangeRule myRule = {abbrev, week, dow, month, hour, offset};
```

Where:

- **abbrev** is a character string abbreviation for the time zone; it must be no longer than five characters.
- **week** is the week of the month that the rule starts.
- **dow** is the day of the week that the rule starts.
- **hour** is the hour in local time that the rule starts (0-23).
- **offset** is the UTC offset in minutes for the time zone being defined.

For convenience, the following symbolic names can be used:

week: First, Second, Third, Fourth, Last **dow:** Sun, Mon, Tue, Wed, Thu, Fri, Sat **month:** Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec

For the Eastern US time zone, the TimeChangeRules could be defined as follows:

```
TimeChangeRule usEDT = {"EDT", Second, Sun, Mar, 2, -240};  //UTC - 4 hours
TimeChangeRule usEST = {"EST", First, Sun, Nov, 2, -300};   //UTC - 5 hours
```

For Central European time zone (Frankfurt, Paris), TimeChangeRules could be as:

```
TimeChangeRule CEST = {"CEST", Last, Sun, Mar, 2, 120};     // Central European
Summer Time
TimeChangeRule CET = {"CET ", Last, Sun, Oct, 3, 60};       // Central European
Standard Time
```

For more information see Timezone info at:
https://github.com/JChristensen/Timezone#readme

Set the Time Zone

Change lines like above in `TinyTrackGPS.cpp` file, at line **111**, with appropriate definition for your time zone.

```
// Central European Time (Frankfurt, Paris) See below for other zone.
TimeChangeRule CEST = {"CEST", Last, Sun, Mar, 2, 120};     // Central European Summer Time
TimeChangeRule CET = {"CET ", Last, Sun, Oct, 3, 60};       // Central European Standard Time
Timezone CE(CEST, CET);
#define TimeZone CE
```

If your time zone is Australia, you can use this lines:

```
// Australia Eastern Time Zone (Sydney, Melbourne)
TimeChangeRule aEDT = {"AEDT", First, Sun, Oct, 2, 660};    // UTC + 11 hours
TimeChangeRule aEST = {"AEST", First, Sun, Apr, 3, 600};    // UTC + 10 hours
Timezone ausET(aEDT, aEST);
#define TimeZone ausET
```

Timezone uses Time library so time is based on the standard Unix time_t. The value is the number of seconds since Jan 1, 1970. And store in unsigned long variable. Arduino Reference describe unsigned long as : `Unsigned long variables are extended size variables for number storage, and store 32 bits (4 bytes). Unlike standard longs unsigned longs won't store negative numbers, making their range from 0 to 4,294,967,295 (2^32 - 1).` It is predict to not be afected with 2038 effect. For more information see Unix Time at Wikipedia, and Year 2038 problem. Assuming that timestamp is 4,294,967,295 (maximal value of unsigned long in Arduino, 2^32 - 1), it wil be in: **GMT: Sunday, 7 February 2106 6:28:15** when time_t overflow. Then time_t reset to 0 and date will be **GMT: Thursday, 1 January 1970 0:00:00**. Visit https://www.epochconverter.com/, an utility for Epoch & Unix Timestamp Conversion.

Platformio

Run command `pio.exe run .`

```
Processing Uno (platform: atmelavr; board: uno; framework: arduino)
------------------------------------------------------------------------------
---------------Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/atmelavr/uno.html
PLATFORM: Atmel AVR (3.4.0) > Arduino Uno
HARDWARE: ATMEGA328P 16MHz, 2KB RAM, 31.50KB Flash
DEBUG: Current (avr-stub) On-board (avr-stub, simavr)
PACKAGES:
 - framework-arduino-avr 5.1.0
 - toolchain-atmelavr 1.70300.191015 (7.3.0)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 14 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <LiquidCrystal> 1.0.7
|-- <TinyGPS> 0.0.0-alpha+sha.db4ef9c97a
|-- <U8g2> 2.28.8
|   |-- <SPI> 1.0
|   |-- <Wire> 1.0
|-- <SdFat> 2.1.0
|   |-- <SPI> 1.0
|-- <LiquidCrystal_I2C> 1.1.4
|   |-- <Wire> 1.0
|-- <Low-Power> 1.81.0
|-- <UTMConversion> 1.0.0
|-- <Timezone> 1.2.4
|   |-- <Time> 1.6.1
|-- <SoftwareSerial> 1.0
Building in release mode
Compiling .pio\build\Uno\src\Display.cpp.o
Compiling .pio\build\Uno\src\TinyTrackGPS.cpp.o
Linking .pio\build\Uno\firmware.elf
Checking size .pio\build\Uno\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=======   ]  68.6% (used 1404 bytes from 2048 bytes)
Flash: [========= ]  85.7% (used 27634 bytes from 32256 bytes)
Building .pio\build\Uno\firmware.hex
================================== [SUCCESS] Took 3.33 seconds
==================================
Environment    Status    Duration
-------------  --------  ------------
Uno            SUCCESS   00:00:03.332
```

```
================================== 1 succeeded in 00:00:03.332
==================================
```

For upload to Arduino use Platformio enviroment or use `platformio.exe run --target upload` command on terminal. This project use LGT_ISP enviroment by default. To burn it use an LGTISP device as describe in LGTISP.

Use of memory (Arduino UNO or equivalent) V0.7 vs. V0.9

NO DISPLAY

Compile V0.7:

```
RAM:   [=====     ]  50.3% (used 1031 bytes from 2048 bytes)
Flash: [=======   ]  70.7% (used 22798 bytes from 32256 bytes)
```

Compile V0.9:

```
RAM:   [======    ]  58.6% (used 1200 bytes from 2048 bytes)
Flash: [========  ]  76.7% (used 24748 bytes from 32256 bytes)
```

LCD 16x2 I2C

Compile V0.7:

```
RAM:   [=======   ]  70.2% (used 1437 bytes from 2048 bytes)
Flash: [========= ]  86.0% (used 27748 bytes from 32256 bytes)
```

Compile V0.9:

```
RAM:   [========  ]  79.6% (used 1631 bytes from 2048 bytes)
Flash: [========= ]  90.2% (used 29100 bytes from 32256 bytes)
```

LCD 16x2 6-WIRED

Compile V0.7:

```
RAM:   [======    ]  59.2% (used 1213 bytes from 2048 bytes)
Flash: [========  ]  81.5% (used 26296 bytes from 32256 bytes)
```

Compile V0.9:

```
RAM:   [=======   ]  68.7% (used 1407 bytes from 2048 bytes)
Flash: [========= ]  85.7% (used 27646 bytes from 32256 bytes)
```

OLED 0'96" 128x64 I2C

Compile V0.7:

```
RAM:   [========  ]  79.7% (used 1632 bytes from 2048 bytes)
Flash: [==========]  96.2% (used 31032 bytes from 32256 bytes)
```

Compile V0.9:

```
RAM:   [========= ]  85.2% (used 1744 bytes from 2048 bytes)
Flash: [==========]  99.8% (used 32190 bytes from 32256 bytes)
```

Changelog

V0.11

- TinyGPS upgrade for NMEA Data Protocol v3.x and GLONASS. Library from https://github.com/fmgomes/TinyGPS (fixed as describe in *TinyGPS library* section.)

V0.10.4

- Fixed SDCard not save.

V0.10

- LowPower library only when no display is defined, to reduce flash memory.
- Connect NMEA 6 GPS module to digital pins 0, 1 (hardware serial). SoftwareSerial library don't use now. So reduce flash memory.
- Fixed TinyGPS library to decode GPRMC *and* GPGGA sentence at same time.
- Better support for LGT8F328P.
- Use LGTISP when use LGT8F328P to burn TinyTrackGPS into board. No bootloader.

- Added lcdgfx library from https://github.com/lexus2k/lcdgfx.
- New logo for splash at start.

V0.9

- Added Timezone library for local time log.

V0.8

- Added UTMConversion library for conversion to UTM coordinates. It has been implemented From library UTMconversion.h (TinyTrackGPS V0.7). Now it is an independent library.

V0.7

- Use Low-Power library to reduce power consumption and gain greater autonomy implementing the project portably using lithium batteries.
- No display support for minimal implementation.
- Fixed some errors when displays on LCD 16x2.
- Fixed error when save log on SD. Sometimes data didn't save correctly.

V0.6

- Fixed error GPS log file when compiling for OLED 0'96".
- Added wait animation for OLED 0'96" 128x64.
- Written new procedure to save data in GPS log file.
- Less global variables, so code with less size when compile it.

V0.5

- Added wait animation for LCD 16x2 on "Waitting for GPS signal..." screen.
- Added support for OLED 0'96" 128x64.
- GPS log file set time for create and modify.
- Use SdFat library, Bill Greiman, for better performance.

- Remove switch for select visual data on LCD 16x2. Now data change automatically every 8 seconds between UTM and grades coordenates.

Working

It works getting info from NMEA module every second and save it into de log file.

Log file Format is:

```
HH:MM:SS,YY.YYYYYY,XX.XXXXXX,ALT,UTM (WGS86)
```

Like this:

```
12:42:47,37.990493,-4.785790,571,30S 343186 4206265
12:42:48,37.990276,-4.785741,571,30S 343190 4206240
12:42:49,37.990062,-4.785705,571,30S 343193 4206216
12:42:50,37.989860,-4.785694,571,30S 343193 4206194
...
```

Where:

- HH - Hours from GPS UTC.
- MM - Minutes.
- SS - Seconds.
- YY.YYYYYY - Degree of latitude.
- XX.XXXXXX - Degree of longitude.
- ALT - Altitude in meters.
- UTM - Coordenates in UTM format(WGS84): Zone Band X Y (00A XXXXXX YYYYYYY)

```
Time,latitude,longitude,alt,utm
06:07:24,38.016925,-4.780960,511,30S 343666 4209189
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:25,38.016914,-4.780968,511,30S 343665 4209188
06:07:33,38.016937,-4.781087,511,30S 343655 4209191
06:07:34,38.016910,-4.781114,511,30S 343653 4209188
06:07:35,38.016899,-4.781131,511,30S 343651 4209187
06:07:36,38.016910,-4.781143,511,30S 343650 4209188
06:07:37,38.016914,-4.781162,511,30S 343649 4209189
06:07:39,38.016891,-4.781203,511,30S 343645 4209186
06:07:40,38.016880,-4.781214,511,30S 343644 4209185
06:07:41,38.016872,-4.781234,511,30S 343642 4209184
```

For conversion to UTM coordinates use UTMConversion library. (https://github.com/RafaelReyesCarmona/UTMConversion)

Example of use:

```
#include "UTMconversion.h"

float flat = 37.8959210;
float flon = -4.7478210;

GPS_UTM utm;

void setup() {
  char utmstr[] = "30S 123456 1234567";

  Serial.begin(9600);

  utm.UTM(flat, flon);
  sprintf(utmstr, "%02d%c %ld %ld", utm.zone(), utm.band(), utm.X(), utm.Y());
  Serial.println(utmstr);
}

void loop() {

}
```

File is named as:

`YYYYMMDD.csv` Example: `20210216.csv`

Where:

- YYYY - Year 4 digits format.
- MM - Mouth.
- DD - Day.

NMEA Secuence from GPS Module.

I am using a Ublox NEO-6MV2 module for get possition and time from GPS System. When GPS signal is ok, the module send the above information through serial:

```
$GPRMC,091620.00,A,3753.16481,N,00447.76212,W,9.209,273.97,201021,,,A*75
$GPVTG,273.97,T,,M,9.209,N,17.064,K,A*03
$GPGGA,091620.00,3753.16481,N,00447.76212,W,1,03,5.72,511.8,M,47.7,M,,*47
$GPGSA,A,2,21,31,22,,,,,,,,,,5.81,5.72,1.00*0F
$GPGSV,2,1,06,01,51,098,32,03,64,025,28,06,14,305,,21,33,114,40*73
$GPGSV,2,2,06,22,46,053,21,31,09,057,37*75
$GPGLL,3753.16481,N,00447.76212,W,091620.00,A,A*78
```

It is very important how to program for get GPS information correctly. (Fixed since V0.10)

With updated GPS module to Ublox NMEA-8M we recibe more information, now the module uses GPS+GLONASS+GALILEO systems. Information through serial port is like above:

```
$GNRMC,102140.00,A,3801.27758,N,00446.88703,W,0.561,,011221,,,A*7E
$GNVTG,,T,,M,0.561,N,1.038,K,A*35
$GNGGA,102140.00,3801.27758,N,00446.88703,W,1,04,3.29,546.4,M,47.8,M,,*55
$GNGSA,A,3,06,30,,,,,,,,,,10.39,3.29,9.86*2D
$GNGSA,A,3,79,71,,,,,,,,,,10.39,3.29,9.86*20
$GPGSV,3,1,09,02,51,274,07,06,45,193,28,07,64,093,,09,39,054,*79
$GPGSV,3,2,09,11,60,261,,13,15,250,06,20,48,313,08,29,04,316,*79
$GPGSV,3,3,09,30,57,167,19*49
$GLGSV,2,1,06,69,24,069,,71,24,294,16,79,30,049,25,80,40,123,*65
$GLGSV,2,2,06,84,06,193,,85,37,245,*68
$GNGLL,3801.27758,N,00446.88703,W,102140.00,A,A*64
```

Information on GPS NMEA sentences

You can get more information about GPS - NMEA sentence information in the web page. Or RF Wireless World page. And SatSleuth Electronic circuits page.

All sentences of NMEA start with "$GX___" secuence. Where X could be 'P', 'L', 'A', 'N'.

- P - GPS satellital system.
- L - GLONASS.
- A - Galileo constellation.
- N - For a combination of 2 or more satellital system.

The above information '_' could be replace with appropiate constellation.

G_RMC sentence (GPRMC, GLRMC, GARMC, GNRMC)

GPRMC secuence is 'Recommended minimum specific GPS/Transit data'

Format is: `$GPRMC,hhmmss.ss,A,LLLL.LLLL,a,YYYYY.YYYY,b,x.xxx,ccc.cc,ddmmyy,v.v,m,M*hh`

- `hhmmss.ss` = UTC time: hh-hour, mm - minute, ss.ss - seconds with decimals.
- `A` = Data status, navigation receiver warning (A=Ok, V=warning)
- `LLLL.LLLL` = Latitude (ddmm.mmmm)
- `a` = North/South ('N' or 'S')
- `YYYYY.YYYY` = Longitude (dddmm.mmmm)
- `b` = East/West = ('E' or 'W')
- `x.xxx` = Speed over ground in knots
- `ccc.cc` = True course in degrees
- `ddmmyy` = UT date (dd-day, mm-mounth, yy-year)
- `v.v` = Magnetic variation degrees (Easterly var. subtracts from true course)
- `m` = East/West of variation ('E' or 'W')
- `M` = Mode (A = Autonomous, D = DGPS, E =DR)
- `*hh` = Checksum

```
eg.: $GPRMC,091620.00,A,3753.16481,N,00447.76212,W,9.209,273.97,201021,,,A*75

  091620.00    UTC Time 09:16:20
  A            Navigation receiver OK
  3753.16481   Latitude 37 deg. 53.16481 min.
  N            North
  00447.76212  Longitude 004 deg. 47.76212 min
  W            West
  9.209        Speed over ground, Knots
```

```
273.97      Course
201021      UTC Date 10 November 2021
A           Mode Autonomous
*75         Checksum
```

G_VTG sentence (GPVTG ... GNVTG)

GPVTG sentence is 'Course and speed information relative to the ground'.

Format is: `$GPVTG,ccc.cc,T,ccc.cc,M,x.xxx,U,ss.sss,K,M*03`

- `ccc.cc` = True course in degrees
- `T` = Reference (T = True heading)
- `ccc.cc` = Course in degrees
- `M` = Reference (M = Magnetic heading)
- `x.xxx` = Speed in knots
- `U` = Units (N = Knots)
- `ss.sss` = Speed in km/h
- `K` = Unit (K = Km/h)
- `M` = Mode (A = Autonomous, D = DGPS, E =DR)
- `*hh` = Checksum

`eg.: $GPVTG,273.97,T,,M,9.209,N,17.064,K,A*03`

G_GGA sentence (GPGGA ... GNGGA)

GPGGA sentence is 'Global positioning system fix data (time, position, fix type data)'.

Format is: `$GPGGA,hhmmss.ss,LLLL.LLLL,a,YYYYY.YYYY,b,P,SS,H.HH,EEE.E,M,GG.G,M,A,ID*hh`

- `hhmmss.ss` = UTC time: hh-hour, mm - minute, ss.ss - seconds with decimals.
- `LLLL.LLLL` = Latitude (ddmm.mmmm)
- `a` = North/South ('N' or 'S')
- `YYYYY.YYYY` = Longitude (dddmm.mmmm)
- `b` = East/West = ('E' or 'W')
- `P` = Position Fix Indicator (0-Unavailable or invalid; 1-GPS SS Mode,valid; 2-Differencial GPS SS Mode,valid; 3-5 Not sopported)
- `SS` = Satellites in use.
- `H.HH` = HDOP (Horizontal Dilution of Precision)

- `EEE.E` = Altitude
- `M` = Unit (M=meters)
- `GG.G` = Geoid Separation
- `M` = Unit (M=meters)
- `A` = Age of difference correction (seconds)
- `ID` = Diff. ref. station ID
- `*hh` = Checksum

```
eg.: $GPGGA,091620.00,3753.16481,N,00447.76212,W,1,03,5.72,511.8,M,47.7,M,,*47
```

```
091620.00    UTC Time 09:16:20
3753.16481   Latitude 37 deg. 53.16481 min.
N            North
00447.76212  Longitude 004 deg. 47.76212 min
W            West
1            GPS SS Mode
03           Satellites in use
5.72         Horizontal Dilution of Precision
511.8        Altitude
M            Unit meters
47.7         Geoid Separation
M            Unit meters
*47          Checksum
```

G_GSA and G_GSV sentences (GPGSA ... GNGSA; GPGSV, GLGSV, GAGSV)

Both are sentences about satellites information. GPGSA sentence is 'Active satellites' and GPGSV is 'Satellites in view'.

GPGLL sentence (G_GLL)

GPGLL sentence is 'Geographic position, latitude, longitude'.

Format is: `$GPGLL,LLLL.LLLL,a,YYYYY.YYYY,b,hhmmss.ss,A,M,*hh`

- `LLLL.LLLL` = Latitude (ddmm.mmmm)
- `a` = North/South ('N' or 'S')
- `YYYYY.YYYY` = Longitude (dddmm.mmmm)
- `b` = East/West = ('E' or 'W')
- `hhmmss.ss` = UTC time: hh-hour, mm - minute, ss.ss - seconds with decimals.
- `A` = Data status, navigation receiver warning (A=Ok, V=warning)

- `M` = Mode (A = Autonomous, D = DGPS, E =DR)
- `*hh` = Checksum

```
eg.: $GPGLL,3753.16481,N,00447.76212,W,091620.00,A,A*78
```

```
  3753.16481   Latitude 37 deg. 53.16481 min.
  N            North
  00447.76212  Longitude 004 deg. 47.76212 min
  W            West
  091620.00    UTC Time 09:16:20
  A            Navigation receiver OK
  A            Mode Autonomous
  *78          Checksum
```

Low Energy Comsuption

`Low-Power` - the library is used to reduce power consumption and gain greater autonomy implementing the project portably using lithium batteries. Use only when no display configuration. Implemented in v0.4 first time and from v0.7.

TinyGPS library

Versions prior V0.10

TinyGPS library works getting information from GPRMC and GPGGA sentences. It extract time, date, latitude, longitude, speed and course information from GPRMC sentence. And altitude, time, latitude, longitude, numbers of satellites in use and hdop information from GPGGA sentence.

The function `bool TinyGPS::encode(char c)` call to `bool TinyGPS::term_complete()` and return `true` when GPRMC or GPGGA sentence is decoded correctly. The above code on TinyTrackGPS:

```
do {
  #ifndef NO_DISPLAY
  LCD.wait_anin(time++);
  #endif
  for (unsigned long start = millis(); millis() - start < 1000;) {
    while (gps_serial.available() > 0) {
      char c = gps_serial.read();
      //Serial.write(c); // uncomment this line if you want to see the GPS data flowing
      if (gps.encode(c)) {// Did a new valid sentence come in?
        gps.crack_datetime(&year_gps, &month_gps, &day_gps, &hour_gps, &minute_gps, &second_gps
, NULL, &age);
        (age != TinyGPS::GPS_INVALID_AGE) ? config = true : config = false;
      }
    }
  }
}while(!config);
```

It is in `setup()` section. This code wait until GPRMC sentence is correctly recieved. Then config time and date whith the code:

```
utctime = makeTime(time_gps);
localtime = TimeZone.toLocal(utctime);
```

Usually NEO6 module is config to 9600 bauds and with 1 Hz for transmit information. So, rest of sentences are ignored in `setup()`.

In `loop()` the while loop get the first sentence from NEO6 module, it is GPRMC sentence.

```
while (gps_serial.available() > 0) {
  char c = gps_serial.read();
  //Serial.write(c); // uncomment this line if you want to see the GPS data flowing
  if (gps.encode(c)) {
    gps.crack_datetime(&year_gps, &month_gps, &day_gps, &hour_gps, &minute_gps, &second_gps,
NULL, &age);
    gps_ok = true;
  }
}
```

To get altitude information is needed to decode GPGAA sentence, so call `GPSRefresh()` to do that.

```
void GPSRefresh()
{
    while (gps_serial.available() > 0)
      gps.encode(gps_serial.read());
}
```

Fixed TinyGPS on V0.10

Now the function `bool TinyGPS::encode(char c)` call to `bool TinyGPS::term_complete()` and return `true` when GPRMC *and* GPGGA sentence is decoded correctly. So all information is decoded at same time. Now `GPSRefresh()` is no neccessary.

Original code TinyGPS:

```
    byte checksum = 16 * from_hex(_term[0]) + from_hex(_term[1]);
    if (checksum == _parity)
    {
      if (_gps_data_good)
      {
#ifndef _GPS_NO_STATS
        ++_good_sentences;
#endif
        _last_time_fix = _new_time_fix;
        _last_position_fix = _new_position_fix;

        switch(_sentence_type)
        {
        case _GPS_SENTENCE_GPRMC:
          _time      = _new_time;
          _date      = _new_date;
          _latitude  = _new_latitude;
          _longitude = _new_longitude;
          _speed     = _new_speed;
          _course    = _new_course;
          break;
        case _GPS_SENTENCE_GPGGA:
          _altitude  = _new_altitude;
          _time      = _new_time;
          _latitude  = _new_latitude;
          _longitude = _new_longitude;
          _numsats   = _new_numsats;
          _hdop      = _new_hdop;
          break;
        }

        return true;
      }
    }
```

Fixed code TinyGPS (change return and break at the end):

```
    byte checksum = 16 * from_hex(_term[0]) + from_hex(_term[1]);
    if (checksum == _parity)
    {
      if (_gps_data_good)
      {
#ifndef _GPS_NO_STATS
        ++_good_sentences;
#endif
        _last_time_fix = _new_time_fix;
        _last_position_fix = _new_position_fix;

        switch(_sentence_type)
        {
        case _GPS_SENTENCE_GPRMC:
          _time      = _new_time;
          _date      = _new_date;
          _latitude  = _new_latitude;
          _longitude = _new_longitude;
          _speed     = _new_speed;
          _course    = _new_course;
          break;
        case _GPS_SENTENCE_GPGGA:
          _altitude  = _new_altitude;
          _time      = _new_time;
          _latitude  = _new_latitude;
          _longitude = _new_longitude;
          _numsats   = _new_numsats;
          _hdop      = _new_hdop;
          return true;
        }


      }
    }
```

Fixed TinyGPS on V0.11

In TinyTrackGPS V0.11, TinyGPS library is a modified version from https://github.com/fmgomes/TinyGPS to adds support for newer NMEA-capable GPS devices that implement the v3.x GNSS spec as well as devices that support GLONASS. This version is fixed to add support to GNGGA sentence and decode G_RMC *and* G_GGA sentences at same time.

Fixed version is placed in 'lib/TinyGPS_GLONASS_fixed'.

```
// the first term determines the sentence type
if (_term_number == 0)
{
  if (!gpsstrcmp(_term, _GPRMC_TERM) || !gpsstrcmp(_term, _GNRMC_TERM))
    _sentence_type = _GPS_SENTENCE_GPRMC;
  else if (!gpsstrcmp(_term, _GPGGA_TERM) || !gpsstrcmp(_term, _GNGGA_TERM))
    _sentence_type = _GPS_SENTENCE_GPGGA;
  else if (!gpsstrcmp(_term, _GNGNS_TERM))
    _sentence_type = _GPS_SENTENCE_GNGNS;
  else if (!gpsstrcmp(_term, _GNGSA_TERM) || !gpsstrcmp(_term, _GPGSA_TERM))
    _sentence_type = _GPS_SENTENCE_GNGSA;
  else if (!gpsstrcmp(_term, _GPGSV_TERM))
    _sentence_type = _GPS_SENTENCE_GPGSV;
  else if (!gpsstrcmp(_term, _GLGSV_TERM))
    _sentence_type = _GPS_SENTENCE_GLGSV;
  else
    _sentence_type = _GPS_SENTENCE_OTHER;
  return false;
}
```

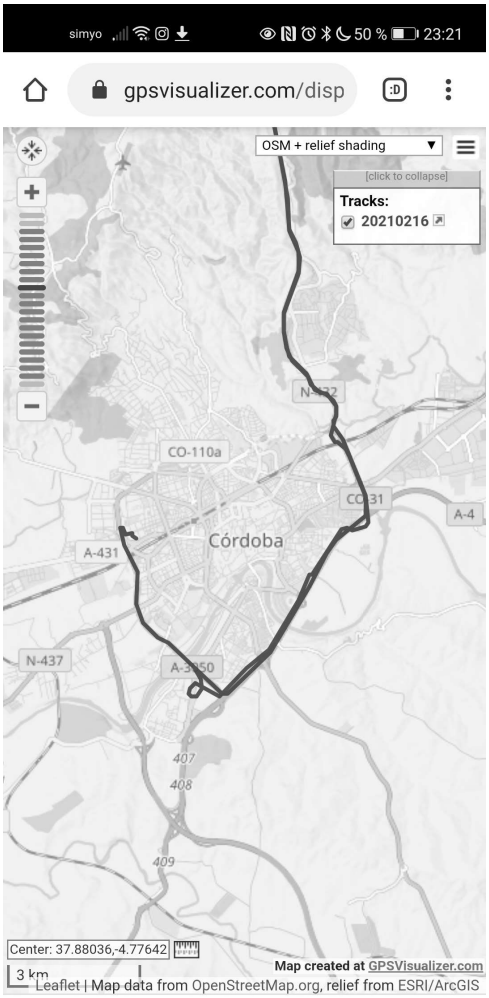
Accuracy

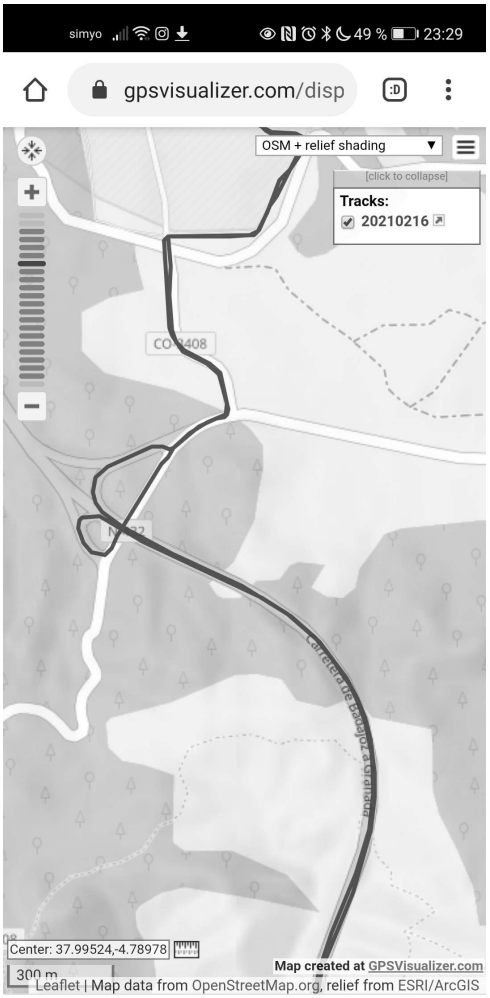
NMEA 6 GPS module accuracy is similar to others GPS devices. In the picture can see it.

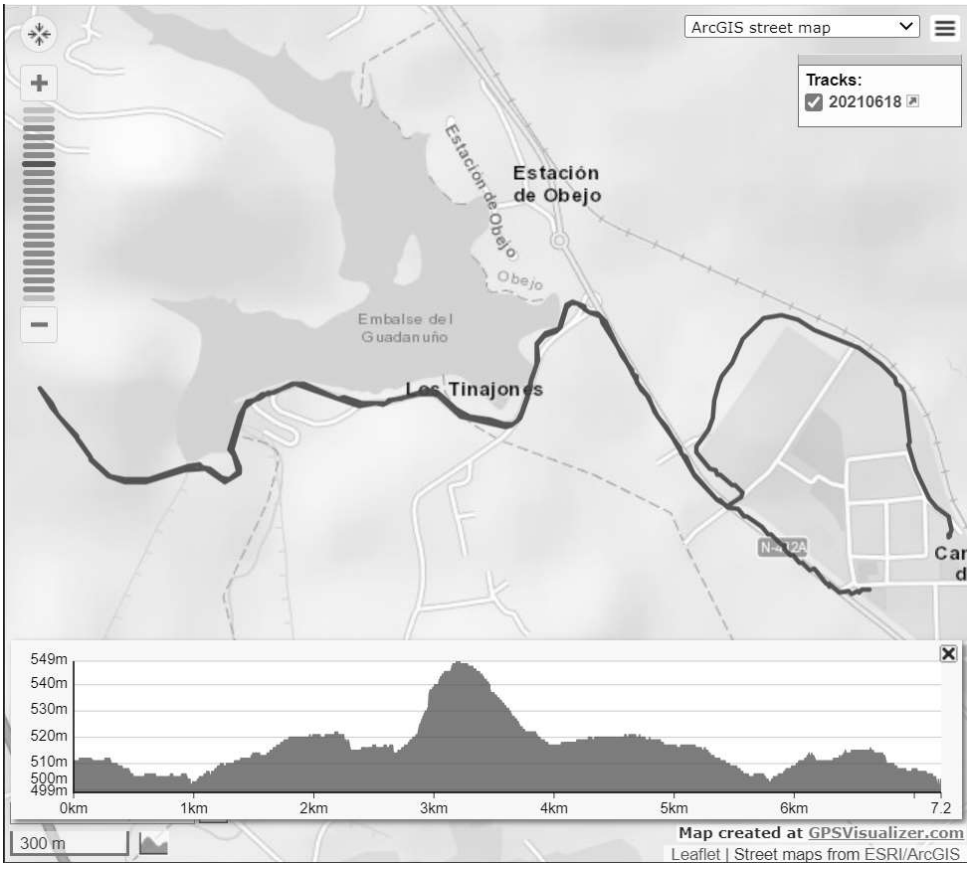
- `Ref` was at *(30S 341554 4194119)* location exactly.
- `TinyGPS` was located at *(30S 341556 4194126)*, 7m error.
- `GPS device` reported *(30S 341553 4194111)*, 8m error.

Draw track on map

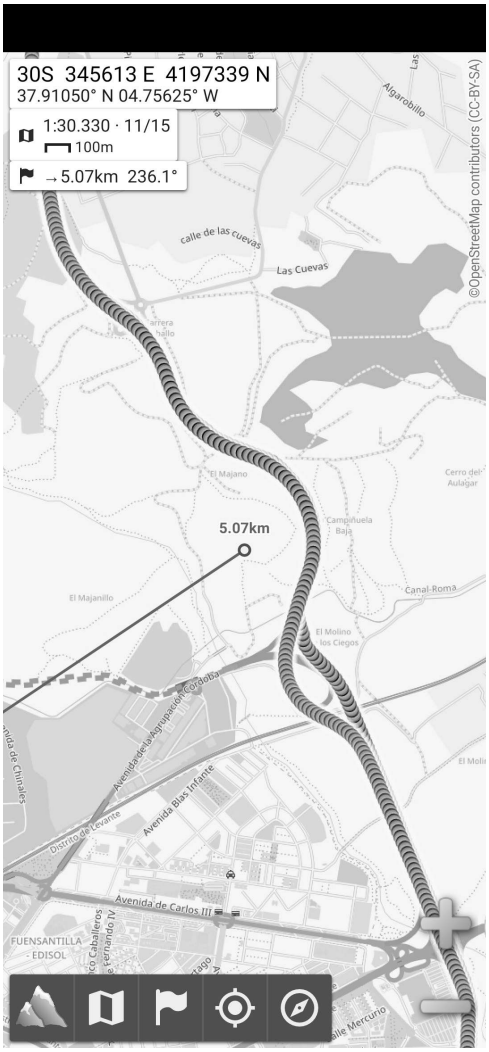
You can upload the file and get the draw on a map using GPS Visualizer.


simyo
50 % 23:21
gpsvisualizer.com/disp
OSM + relief shading
[click to collapse]
Tracks:
20210216
CO-110a
A-4
Córdoba
A-431
N-437
Center: 37.88036,-4.77642
3 km
Map created at GPSVisualizer.com
Leaflet | Map data from OpenStreetMap.org, relief from ESRI/ArcGIS


simyo
49 % 23:29
gpsvisualizer.com/disp
OSM + relief shading
[click to collapse]
Tracks:
20210216
Center: 37.99524,-4.78978
300 m
Map created at GPSVisualizer.com
Leaflet | Map data from OpenStreetMap.org, relief from ESRI/ArcGIS


ArcGIS street map
Tracks:
20210618
Estación de Obejo
Embalse del Guadanuño
Los Tinajones
549m
540m
530m
520m
510m
500m
499m
0km
1km
2km
3km
4km
5km
6km
7.2
300 m
Map created at GPSVisualizer.com
Leaflet | Street maps from ESRI/ArcGIS

Or using apps like AlpineQuest.



SdFat Library

SdFat library provides an SPI interface to connect an SD card module with any microcontroller which supports this communication interface. MicroSD module use SPI communication interface to connect with microcontrollers. Using a micro SD card becomes very handy for applications where we need to store files or any data as this project.

We will be able to read and write data to and from SD cards through the SPI communication protocol with the help of this library. There are different types of microSD card modules available in the market, but the most common pinout configuration has 6 terminals consisting of SPI and power supply terminals:

Pin Name	Description

Pin Name	Description
GND	This is the ground pin which should be connected with the ground pin of Arduino.
VCC	This pin supplies power to the module. The power supply of ~4.5V-5.5V. The adapter consists of a 3.3V voltage regulator circuit as well. It is connected with 5V pin of Arduino.
CS	This is the Chip Select pin for SPI communication.
MOSI	This is called the 'Master Out Slave In.' It is used as the SPI input to the module.
SCK	This is called the 'Serial Clock' pin which is used in SPI serial clock output.
MISO	This is called the 'Master in Slave Out.' It is used as the SPI output from the module.

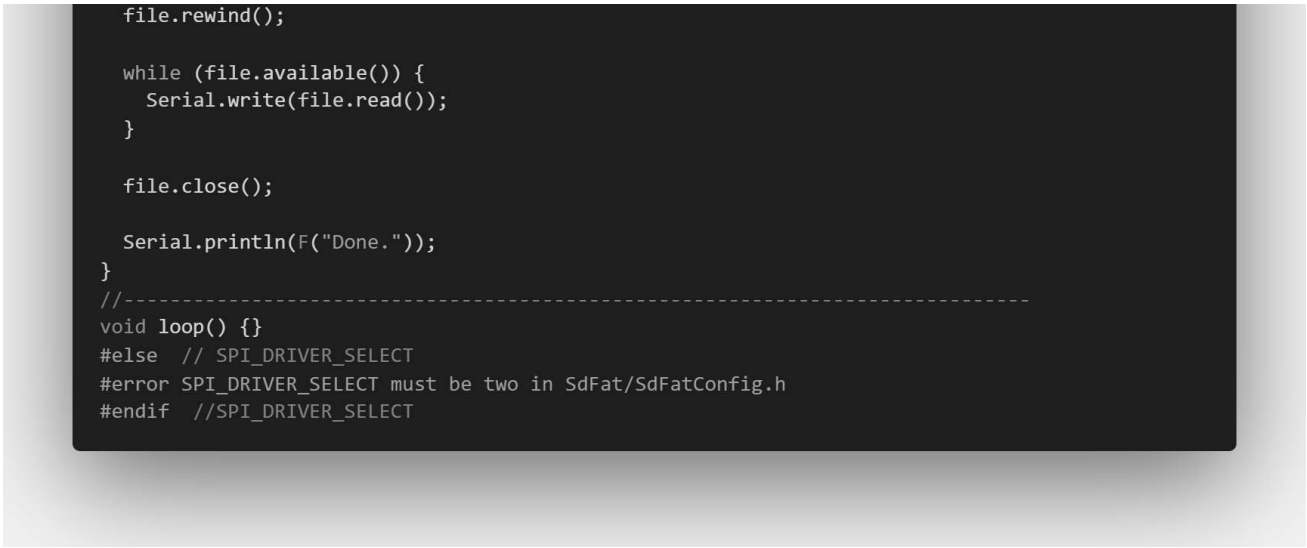
This page has more information about Micro SD Card Interfacing with Arduino.

SdFat library, Bill Greiman, used externat SPI driver config 'SdFatConfig.h' as:

```
//------------------------------------------------------------------------------
// Driver options
/**
 * If the symbol SPI_DRIVER_SELECT is:
 *
 * 0 - An optimized custom SPI driver is used if it exists
 *     else the standard library driver is used.
 *
 * 1 - The standard library driver is always used.
 *
 * 2 - An external SPI driver of SoftSpiDriver template class is always used.
 *
 * 3 - An external SPI driver derived from SdSpiBaseClass is always used.
 */
#ifndef SPI_DRIVER_SELECT
#if defined(__LGT8F__) || defined(__AVR_ATMEGA328P__)
#define SPI_DRIVER_SELECT 2
#else
#define SPI_DRIVER_SELECT 0
#endif
#endif  // SPI_DRIVER_SELECT
```

I used SoftwareSPI driver as you can see in the example code 'SoftwareSPI.ino':

```
// An example of the SoftSpiDriver template class.
// This example is for an old Adafruit Data Logging Shield on a Mega.
// Software SPI is required on Mega since this shield connects to pins 10-13.
// This example will also run on an Uno and other boards using software SPI.
//
#include "SdFat.h"
#if SPI_DRIVER_SELECT == 2  // Must be set in SdFat/SdFatConfig.h

// SD_FAT_TYPE = 0 for SdFat/File as defined in SdFatConfig.h,
// 1 for FAT16/FAT32, 2 for exFAT, 3 for FAT16/FAT32 and exFAT.
#define SD_FAT_TYPE 0
//
// Chip select may be constant or RAM variable.
const uint8_t SD_CS_PIN = 10;
//
// Pin numbers in templates must be constants.
const uint8_t SOFT_MISO_PIN = 12;
const uint8_t SOFT_MOSI_PIN = 11;
const uint8_t SOFT_SCK_PIN  = 13;

// SdFat software SPI template
SoftSpiDriver<SOFT_MISO_PIN, SOFT_MOSI_PIN, SOFT_SCK_PIN> softSpi;
// Speed argument is ignored for software SPI.
#if ENABLE_DEDICATED_SPI
#define SD_CONFIG SdSpiConfig(SD_CS_PIN, DEDICATED_SPI, SD_SCK_MHZ(0), &softSpi)
#else  // ENABLE_DEDICATED_SPI
#define SD_CONFIG SdSpiConfig(SD_CS_PIN, SHARED_SPI, SD_SCK_MHZ(0), &softSpi)
#endif  // ENABLE_DEDICATED_SPI

#if SD_FAT_TYPE == 0
SdFat sd;
File file;
#elif SD_FAT_TYPE == 1
SdFat32 sd;
File32 file;
#elif SD_FAT_TYPE == 2
SdExFat sd;
ExFile file;
#elif SD_FAT_TYPE == 3
SdFs sd;
FsFile file;
#else  // SD_FAT_TYPE
#error Invalid SD_FAT_TYPE
#endif  // SD_FAT_TYPE

void setup() {
  Serial.begin(9600);
  // Wait for USB Serial
  while (!Serial) {
    yield();
  }
  Serial.println("Type any character to start");
  while (!Serial.available()) {
    yield();
  }

  if (!sd.begin(SD_CONFIG)) {
    sd.initErrorHalt();
  }

  if (!file.open("SoftSPI.txt", O_RDWR | O_CREAT)) {
    sd.errorHalt(F("open failed"));
  }
  file.println(F("This line was printed using software SPI."));
```



License

Authors

Copyright © 2019-2021 Francisco Rafael Reyes Carmona. Contact me: rafael.reyes.carmona@gmail.com

Credits

Compass icon at the beginning is from Flaticon.es designed by DinosoftLabs and licensed by free license.